MSafe

Audit Report

MOVEBIT

contact@movebit.xyz

https://twitter.com/movebit_

Thu Oct 19 2023

MSafe Audit Report

1 Executive Summary

1.1 Project Information

Description	The first multi-signature, non-custodial digital assets management solutions built on Move
Type	Asset Management
Auditors	MoveBit
Timeline	Fri Sep 22 2023 - Thu Oct 19 2023
Languages	Move
Platform	Sui
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/Momentum-Safe/mpay-core-sui
Commits	dab3565fcc75b5f352279d405e8e628219f4fb2c 74cb692ebbd1c4d9e67d825f0c0fcb368eb6310d abddd2264463fce6442b81306f4853ddc00926f4

1.2 Files in Scope

The following are the SHA1 hashes of the original reviewed files.

ID	File	SHA-1 Hash
MOV	move/Move.toml	e2d4d669483729ec879fc39f3f1fe6d645a23558
ROL	move/sources/role.move	2d8c8d364d98eca0d80f783fef811f9b4c425310
TUT	move/sources/test_utils.move	903b840932703c34ceb1f361021b2c8459afa03a
STE	move/sources/stream_test.move	c7f80cf577e02747d8ac19c740cd0cca4cea8461
VAU	move/sources/vault.move	3217b6e2e09be3af305546501bf7d1fd0c9f752e
STR	move/sources/stream.move	bf55c49854136a48ed8bc6527ed4d0e4a273afa3
FMO	move/sources/fee_module.move	663cbf88ac3271d566f0e14a020fbe726158890e

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	1	1	0
Informational	0	0	0
Minor	1	1	0
Medium	0	0	0
Major	0	0	0
Critical	0	0	0

1.4 MoveBit Audit Breakdown

MoveBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow by bit operations
- Number of rounding errors
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting
- Unchecked CALL Return Values
- The flow of capability
- Witness Type

1.5 Methodology

The security team adopted the **"Testing and Automated Analysis"**, **"Code Review"** and **"Formal Verification"** strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

(1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

(2) Code Review

The code scope is illustrated in section 1.2.

(3) Formal Verification

Perform formal verification for key functions with the Move Prover.

(4) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

2 Summary

This report has been commissioned by MSafe to identify any potential issues and vulnerabilities in the source code of the MSafe smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

During the audit, we identified 1 issues of varying severity, listed below.

ID	Title	Severity	Status
STR-1	Unused Constant	Minor	Fixed

3 Participant Process

Here are the relevant actors with their respective abilities within the MSafe Smart Contract:

Admin

- Admin can set the `create_fee_numerator` through `set_streaming_fee()`.
- Admin can set the `create_flat_fee` through `set_streaming_flat_fee()`.
- Admin can set the `claim_fee_numerator` through `set_claim_fee()`.
- Admin can set the fee receiver through `set_collector()`.
- Admin can transfer administrative privileges to another account through `transfer_admin()`.

Collector

- Collector has the ability to withdraw fees through `withdraw_fee()`.

Creator

- Creator can create a `Stream` through `create_stream()`.
- Creator can cancel the `Stream` through `cancel_stream()`.

Receiver

- Receiver can set whether they allow other accounts to execute claims on their behalf through `set_auto_claim()`.
- Receiver can claim the funds through `claim_stream()`.

User

- Other accounts can help the recipient perform the collection operation through `claim_stream_by_proxy()`.

4 Findings

STR-1 Unused Constant

Severity: Minor

Status: Fixed

Code Location:

move/sources/stream.move#51

Descriptions:

The main consequence of the Unused Constants defect is the increase in gas costs during module deployment, leading to gas wastage.

```
const SECOND: u64 = 1000;
```

Suggestion:

It is recommended to delete unused constants.

Resolution:

This issue has been fixed. The client deleted unused constants.

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed:** The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your own risk. A report does not imply an endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.